

OSDA API

Offline Shot Data Analysis Application Programmer Interface

Introduction

What is the OSDA API?

A set of Java classes to read the SDA data base and one class to read the Lumberjack values.

What can you do with it?

1. Get the value and time stamp of an ACNET variable for one store.
2. Get the value and time stamp of several variables and their time stamp for one store
3. Get the value and time stamp of one variable in several stores with or without a condition
4. Get the value of a D44 (Lumberjack) for a specified time frame.

Contents

- Downloading and Installing the OSDA API
- Summary of the classes
- Get a value and time stamp for a variable in one store
- Get the value and time stamp of several variables in one store
- Get the value and time stamp of one variable in several stores with a condition
- Error handling
- Get the value of a D44 variable over a time range.

Downloads

1. If you do not have JBuilder, download the free personal copy from from:
<http://www.borland.com/products/downloads/>
2. Download the jdom-b8 and xerces jars from:
<http://www-bd.fnal.gov/javaapplications/jars/jdom-b8.jar>
<http://www-bd.fnal.gov/javaapplications/jars/xerces.jar>
1. Download the osda.jar from:
<http://www-bd.fnal.gov/javaapplications/jars/split/osda.jar>

Installation

- In JBuilder, select Project -> Project Properties and click on the Required Libraries tab.
- Add the following jar files, listing the full path
 - jdom-b8.jar
 - xerces.jar
 - osda.jar

OSDA Classes

- Store – represents an SDA Store
- Case – represents an SDA case with owner and case name
- Variable – an ACNET device
- VarInStores – an ACNET device across several stores
- ValidStore – a vector of stores
- DBError – information about an SDA data base error
- D44Variable – a D44 variable over a specified time
- Utilities – to read and send the XML, and some basic statistics.
- Tester – examples and tests

The Store Class

Store

- Represents an SDA store.
- A store constructor takes an integer or a String representing the store number
`Store s = new Store(1396);`
- The main method for this class is **getvalueOfVariable**, which returns a Variable filled with the values for this store.
- Only implemented for the Collider Shot owner.

The Case Class

Case

- Represents an SDA case.
- Create a case with an owner and case name.

```
// create a case
Case hep =new Case("ColliderShot","HEP");
```

The Variable Class

Variable

- Represents an ACNET variable for a given case.
- Can be: a scalar, a single value, a scalar in a set, an array, an array in a set
- Create a variable with a device name and case

```
Variable myVar =
new Variable(hep, "C:FBIANG");
```

1. Get the value of a variable for one store

```
// create a case
Case hep = new Case("ColliderShot", "HEP");
// create a variable
Variable myVar = new Variable(hep, "C:FBIANG"));
// create a store
Store myStore = new Store(1396);
// retrieve a variable from the store
Variable var = myStore.getValueOfVariable(myVar);
// print the values of the variable
var.printDebugInfo("");
```

Result for (1396, HEP, C:FBIANG)

...

Set = 0, index = 37 is: 0.0

Set = 1, index = 0 is: 246.71187257766724

Set = 1, index = 1 is: 9.950951255857944

...

Set = 55, index = 35 is: 2.8986256308853626

Set = 55, index = 36 is: 2.5997980758547783

...

Get the value and time of a variable for one store cont.

```
// fetch value the set and index
double value =
    var.getOneValueArrayInSet(55,36);
// print it
System.out.println("The value for " +
    var.name + " in case " + var.scase.name
    + " is: " + value);
// get the timestamp in ms.
long time = var.ADBB.getDateInASet(55);
```

Result:

The value for C:FBIANG in case HEP is:
-3.7653815411031246

Get values for sets, arrays, and scalars

double getOneValueArray(int index)

Get a value for for a particular array index

double getOneValueInSets(int iSet)

Get a value for a scalar in a particular set

double getVal()

Get the value of a scalar

2. Get the value of several variables for one store

```
Variable[] theVars = new Variable[3];
// Array in sets
theVars[0] = new Variable(new
    Case("ColliderShot", "HEP"), "C:B0ILUM");
// Array
theVars[1] = new Variable(new Case( "ColliderShot",
    "Before Ramp"), "C:FBIANG");
// Scalar in Set
theVars[2] = new Variable(new Case( "ColliderShot",
    "Inject Pbars"), "A:IBEAMB");
Store myStore = new Store(1396);
myStore.getValueOfVariables(theVars);
double value = theVars[0].getOneValueArrayInSet(50,30);
```

3. Get the value of one variable in several stores with a condition

Use the class: **VarInStores**

- The values of an ACNET variable in one or more stores.
- This class has two data members:
 - a ValidStores (a list of stores)
 - a Variable
- It has multiple constructors for convenience.
For instance, if the user needs only the data for one store, he can give the unique store number.

Example: One variable in several stores with a condition

```
// build a list of stores
ValidStores aValidStores = new ValidStores(1172, 1183);
// create a condition variable
Variable conditionVar = new Variable(InjPB, "C:FBIPNG");
aValidStores.addCondition(conditionVar,3000,8000,1,0);
// create a variable to find
Variable findVar = new Variable (InjPB, "C:FBIPNG");
// find the values of the variable in the stores
// note: constructor also populates the VarInStores
VarInStores myVar = new VarInStores(findVar,
    aValidStores);
// get the value out, specify store, set, and index.
double value = myVar.getOneValueArrayInSet(1172,55,
    30);
```

Error Checking

- **DBError** This class contains the pertinent information when an error occurs while parsing the SDA XML. The Variable class has a DBError data member. To check a variable for an error:

```
Variable avar = aStore.getValueOfVariable(v);  
// check if there was an error reading the values  
// from the SDA data base  
if (avar.hasError()) {  
    avar.printError();  
} else System.out.println("OK - no errors");
```

4. Get the value of a D44 variable over a time range.

```
// create a new D44 variable
D44Variable var = new D44Variable();
// decide on the start and end date and fetch the
// values on line for this time
java.text.SimpleDateFormat sdf = new
    java.text.SimpleDateFormat
    ("MM/dd/yyyy hh:mm:ss(fff)");
Date start = sdf.parse("05/31/2002 17:45:00(376)");
Date stop = sdf.parse("06/01/2002 17:45:00(376)");
var.fetchOnLineData(start,stop,"M:OUTTMP","ArkIV");
```

D44 Example—continued

```
// Print the first value, time, and relative time
System.out.println("The first value is: " +
    var.getOneValue(0));
System.out.println("The time of the first value
    is: " + var.getTime(0));
System.out.println("The relative time of the
    first value is: " + var.getRelativeTime(0));
// print all times and values to the console and
// to a file
boolean relativeTime = true;var.printTable("", 
    relativeTime); // to console
var.printTable("C:/SDA/test/Test1/d44table.txt",
    relativeTime); // to file
```

Examples and Utilities

- **Tester** Examples and tests for the osda package.
- **Utilities** The Utilities for basic statistics and for sending and parsing XML.

More help

Documentation:

[http://www-
bd.fnal.gov/javadoc/build/api/gov/fNAL/contr
ols/applications/osda/package-
summary.html](http://www-bd.fnal.gov/javadoc/build/api/gov/fNAL/controls/applications/osda/package-summary.html)